Computer Vision

Convolutional Neural Networks

Luis F. Teixeira | M.EIC | FEUP

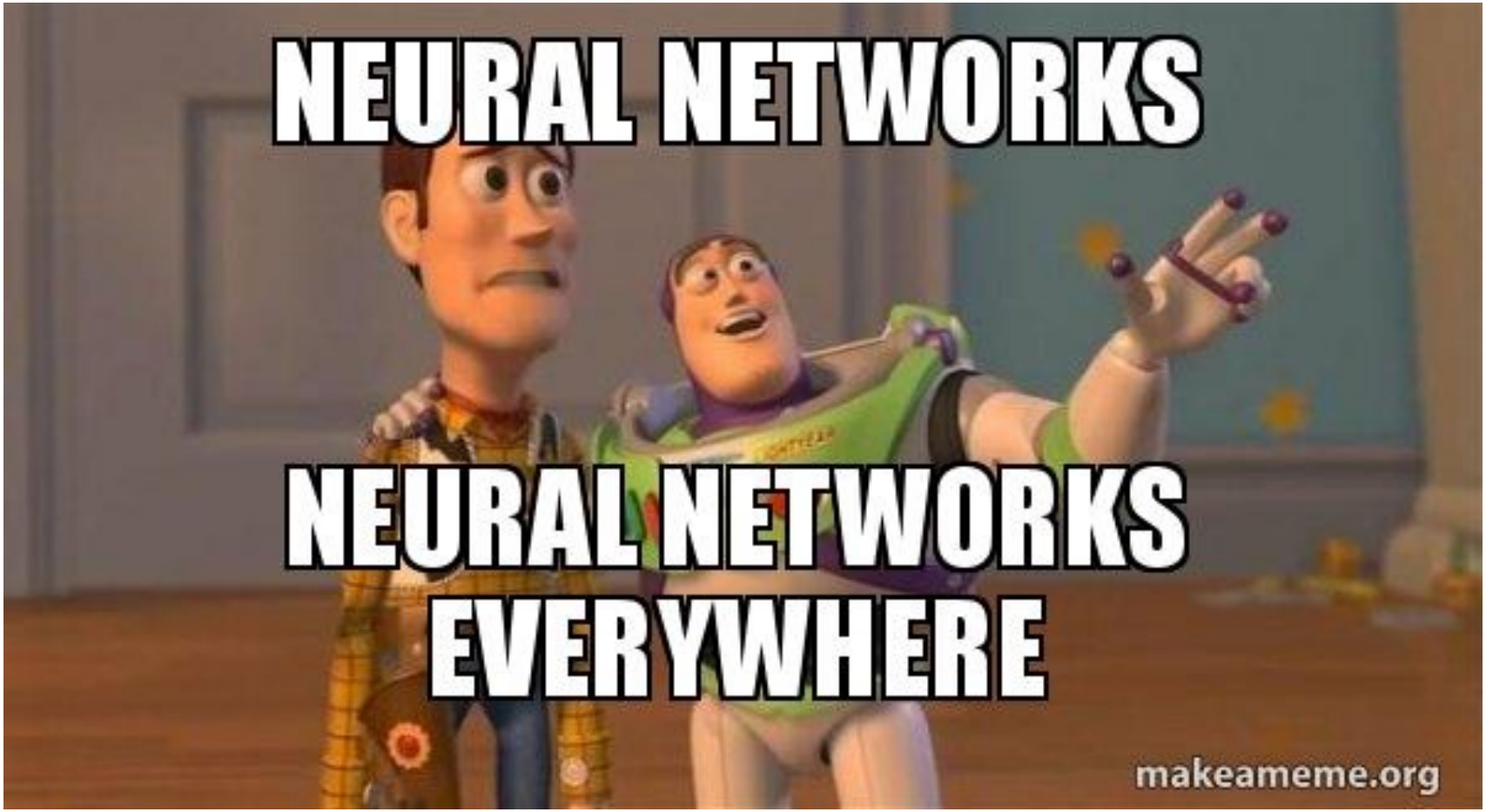
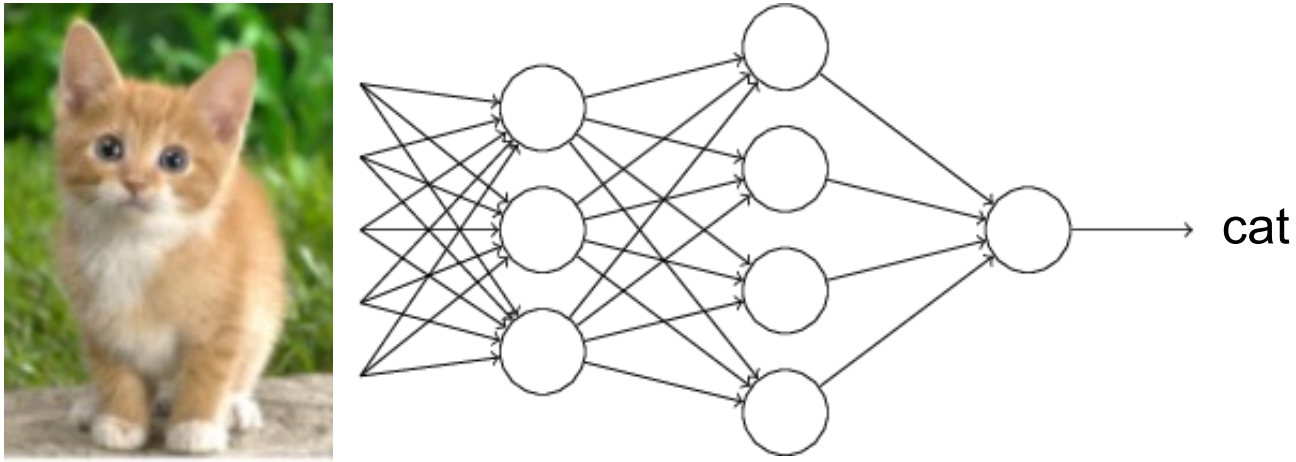


Image source: https://dudeperf3ct.github.io/

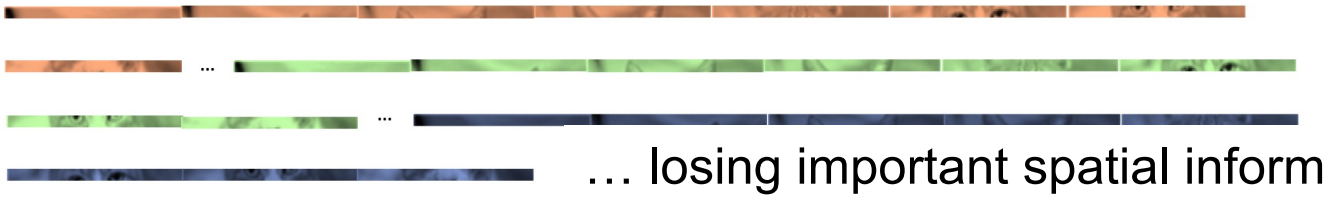
Using MLPs with Images



Can we make this work?

Using MLPs with Images

- Image needs to be represented as a long vector...



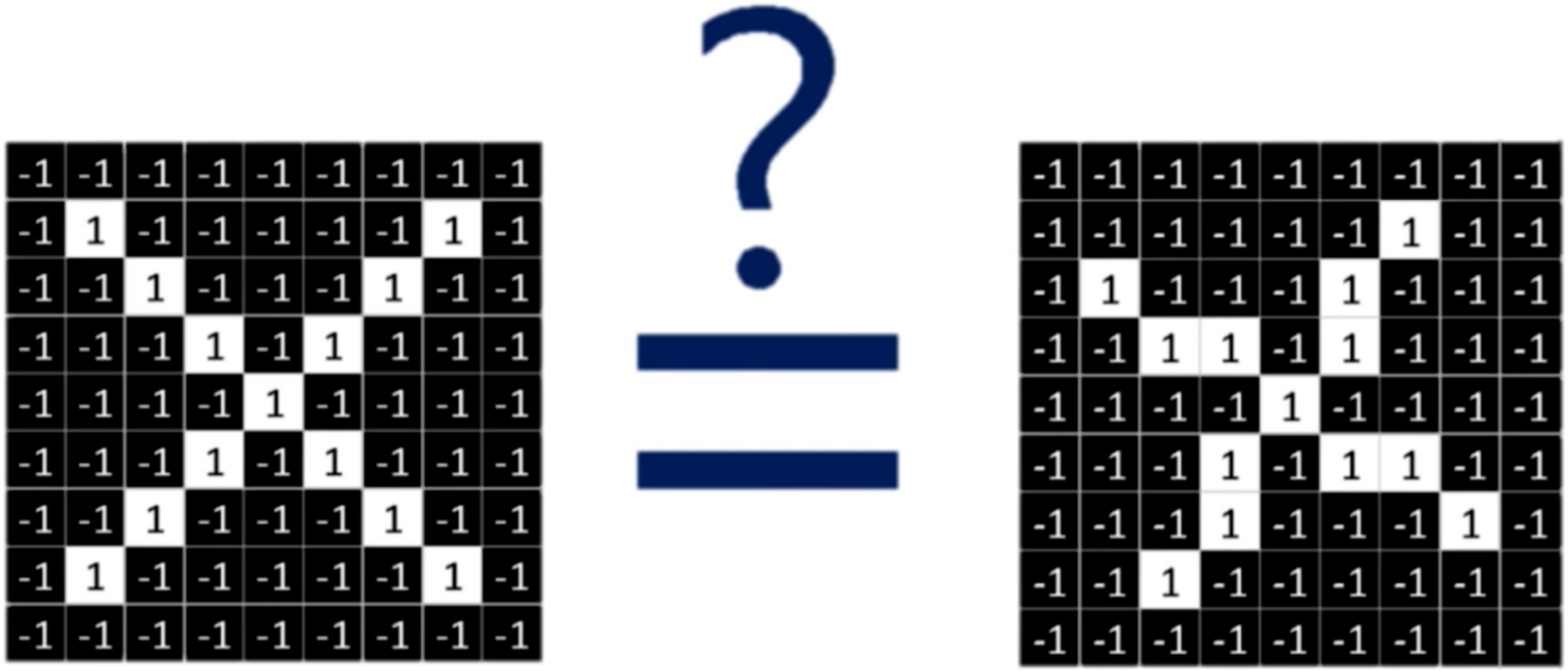
… losing important spatial information.

- A weight vector that learns to match a specific feature in cat images will not match the same feature in the same but shifted or scaled image

  → The MLP is not invariant to image translation or scaling

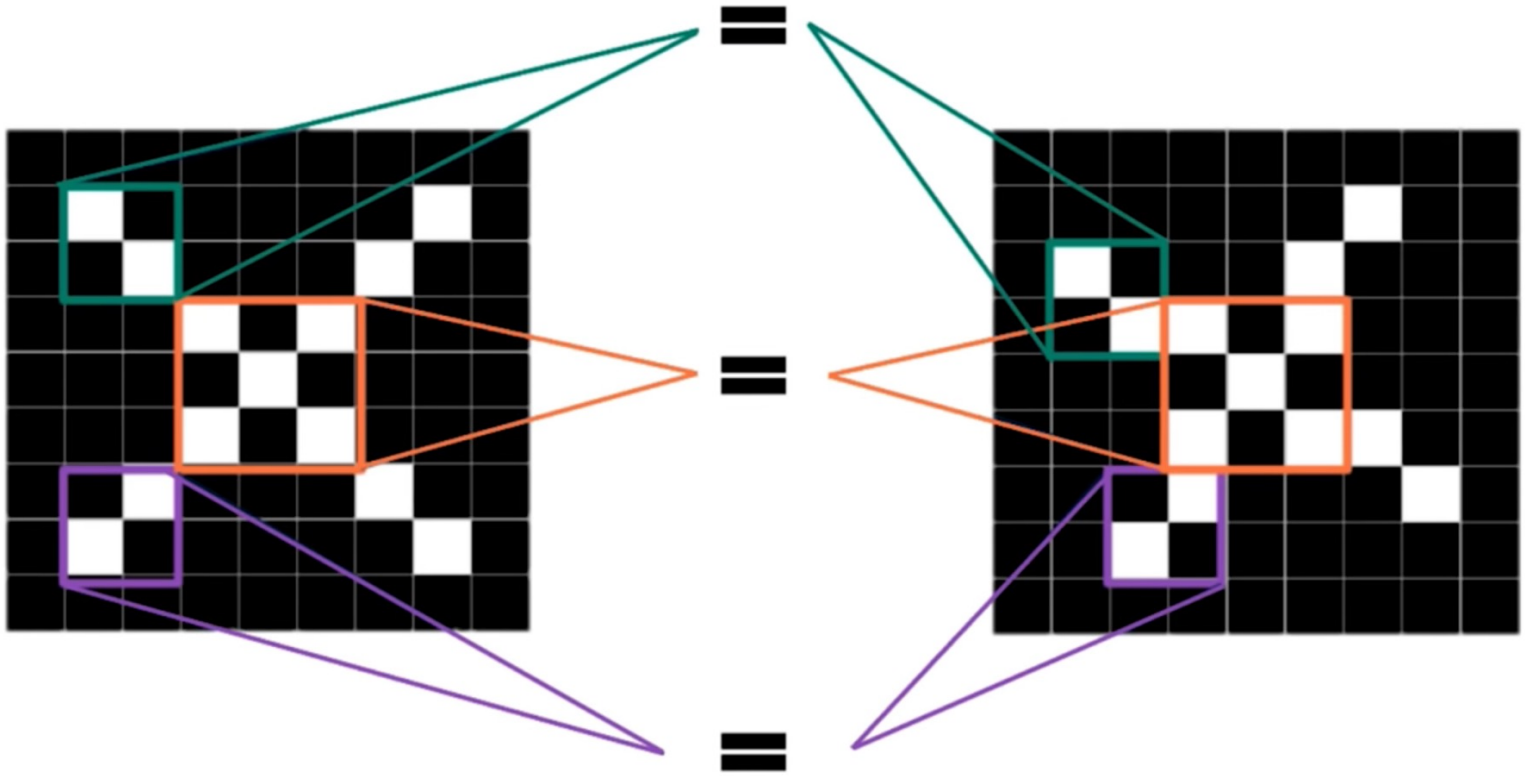
- Also, the MLP model has many weights that are not learning anything useful for images

Interpreting Images



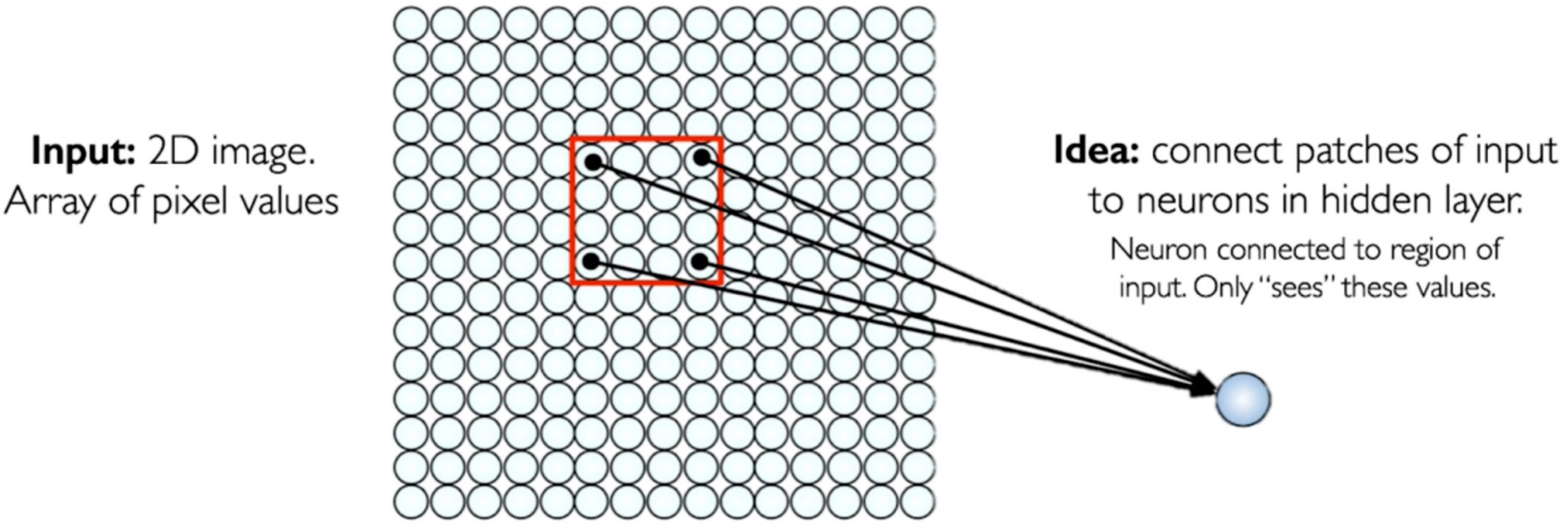
We want to be able to classify an object in the image, even if shifted, resized, rotated, deformed.

Interpreting Images



Instead of comparing the whole image, we should compare **local features**

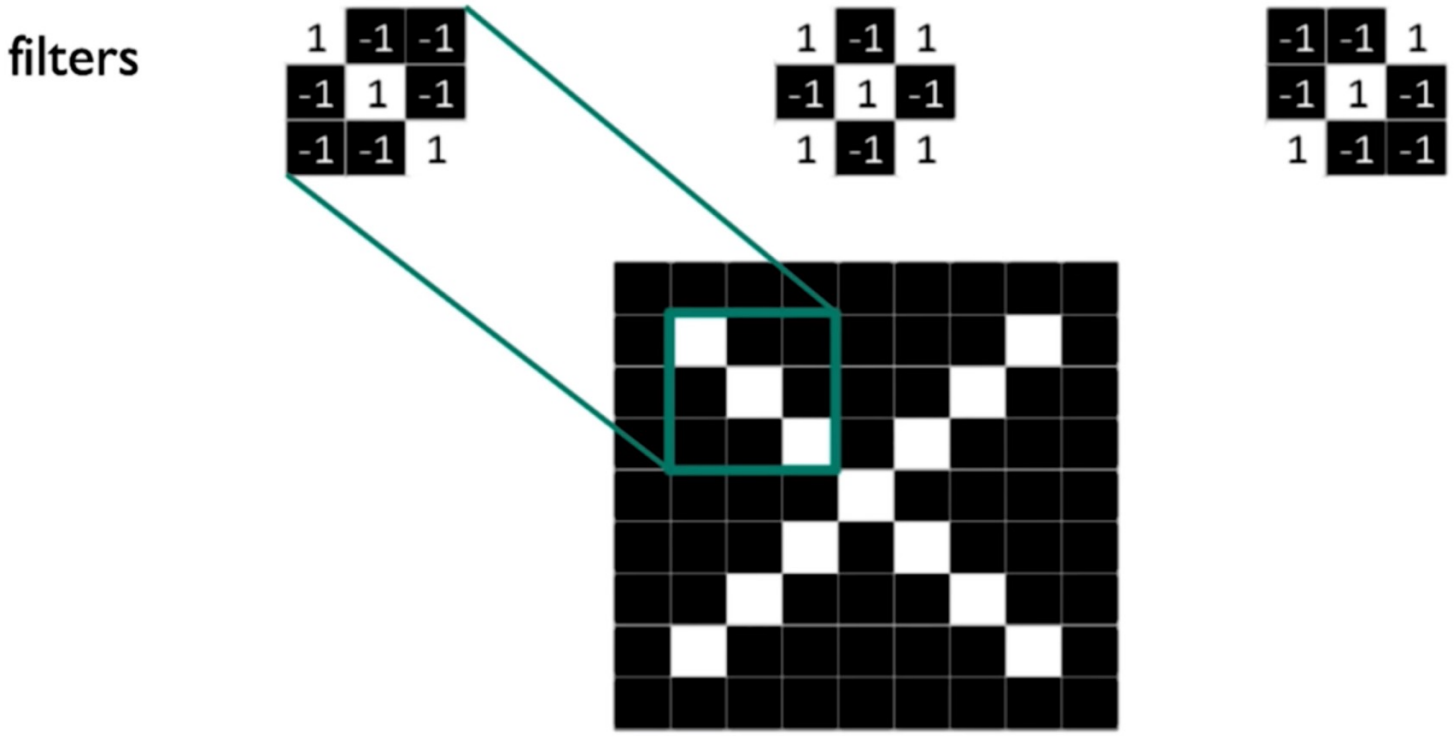
Using Spatial Structure



But, how do we analyse the complete image?

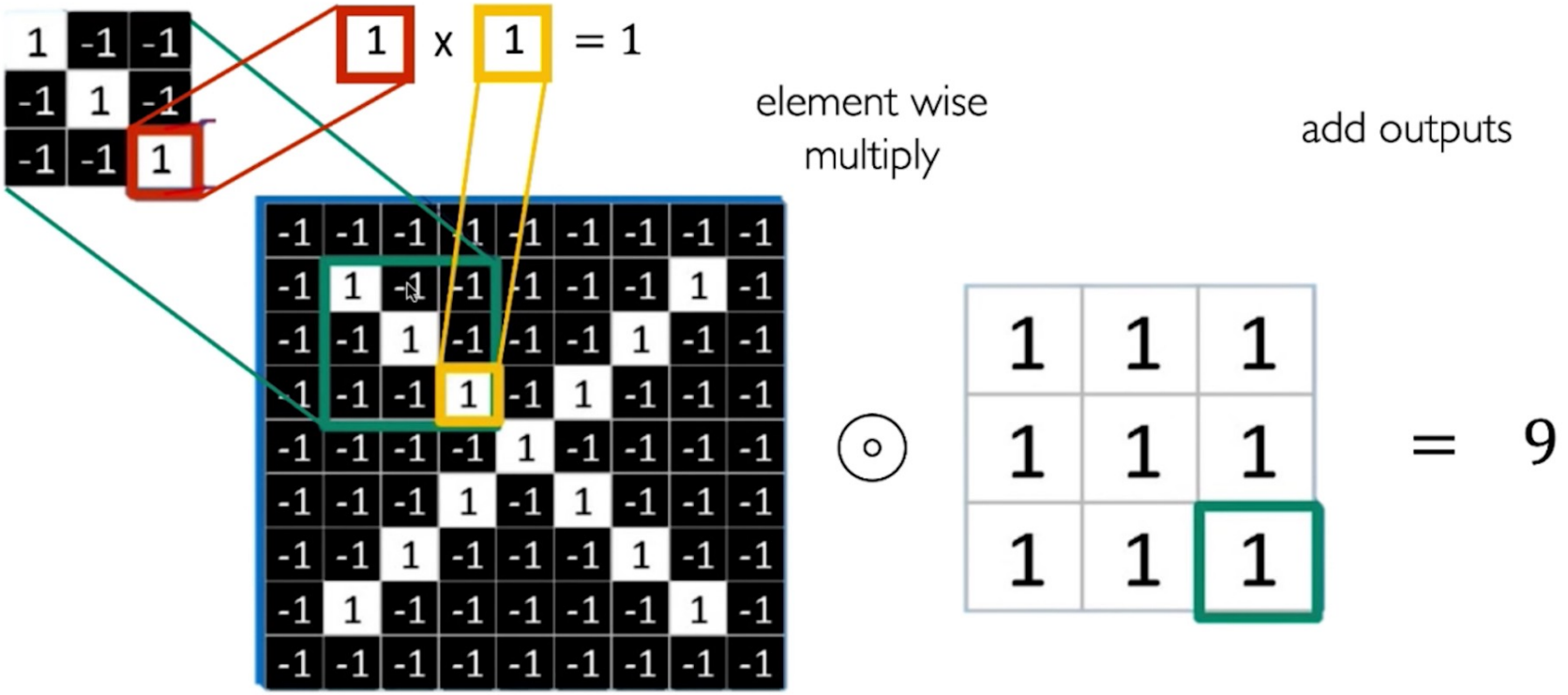
- Multiple neurons with shared weights
- In practice, we use a sliding window to go through the image

Feature Extraction with Convolution



The features are extracted using filters based on a **convolution operation**

Convolution



Convolution

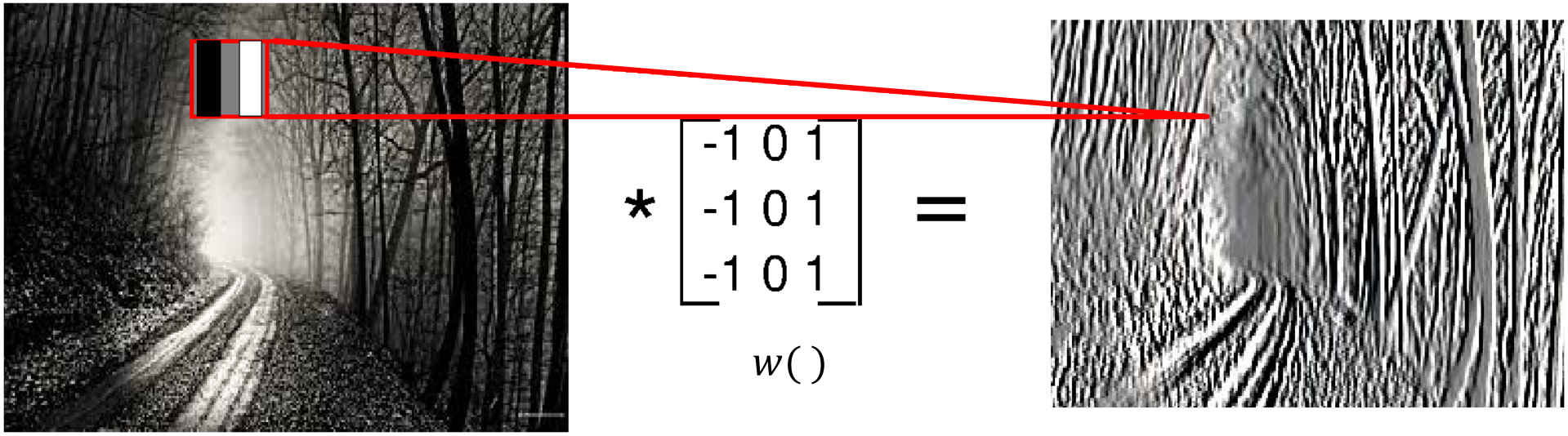
- An example

$$x = \begin{bmatrix} 3 & 3 & 2 & 1 & 0 \\ 0 & 0 & 1 & 3 & 1 \\ 3 & 1 & 2 & 2 & 3 \\ 2 & 0 & 0 & 2 & 2 \\ 2 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad w = \begin{bmatrix} 0 & 1 & 2 \\ 2 & 2 & 0 \\ 0 & 1 & 2 \end{bmatrix}$$

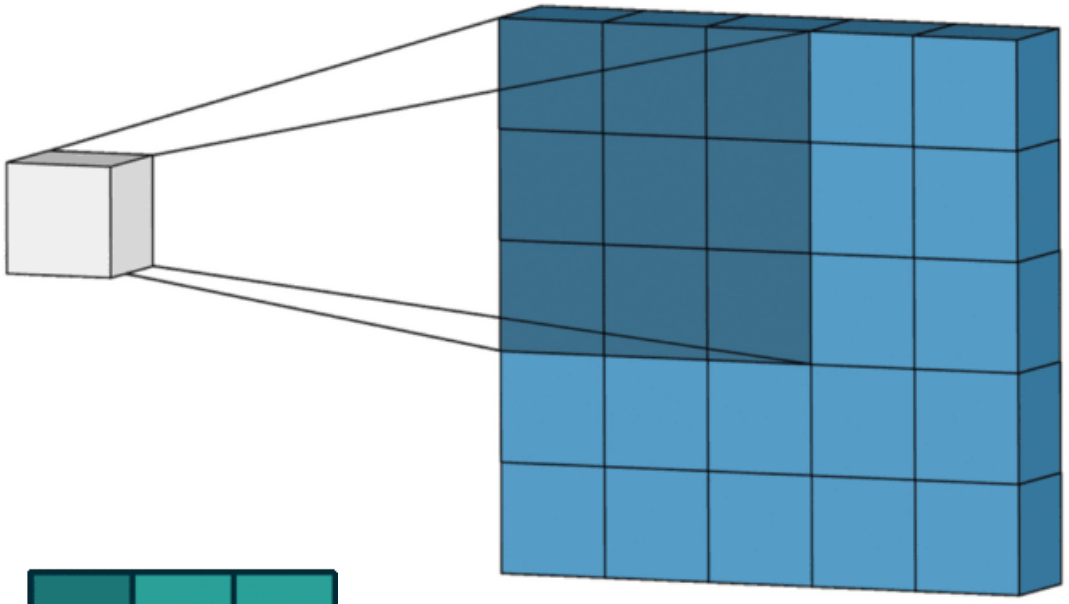
$$y = x * w = \begin{bmatrix} \quad ? \quad \end{bmatrix}$$

Convolution

Edge Detection



Convolution Layer



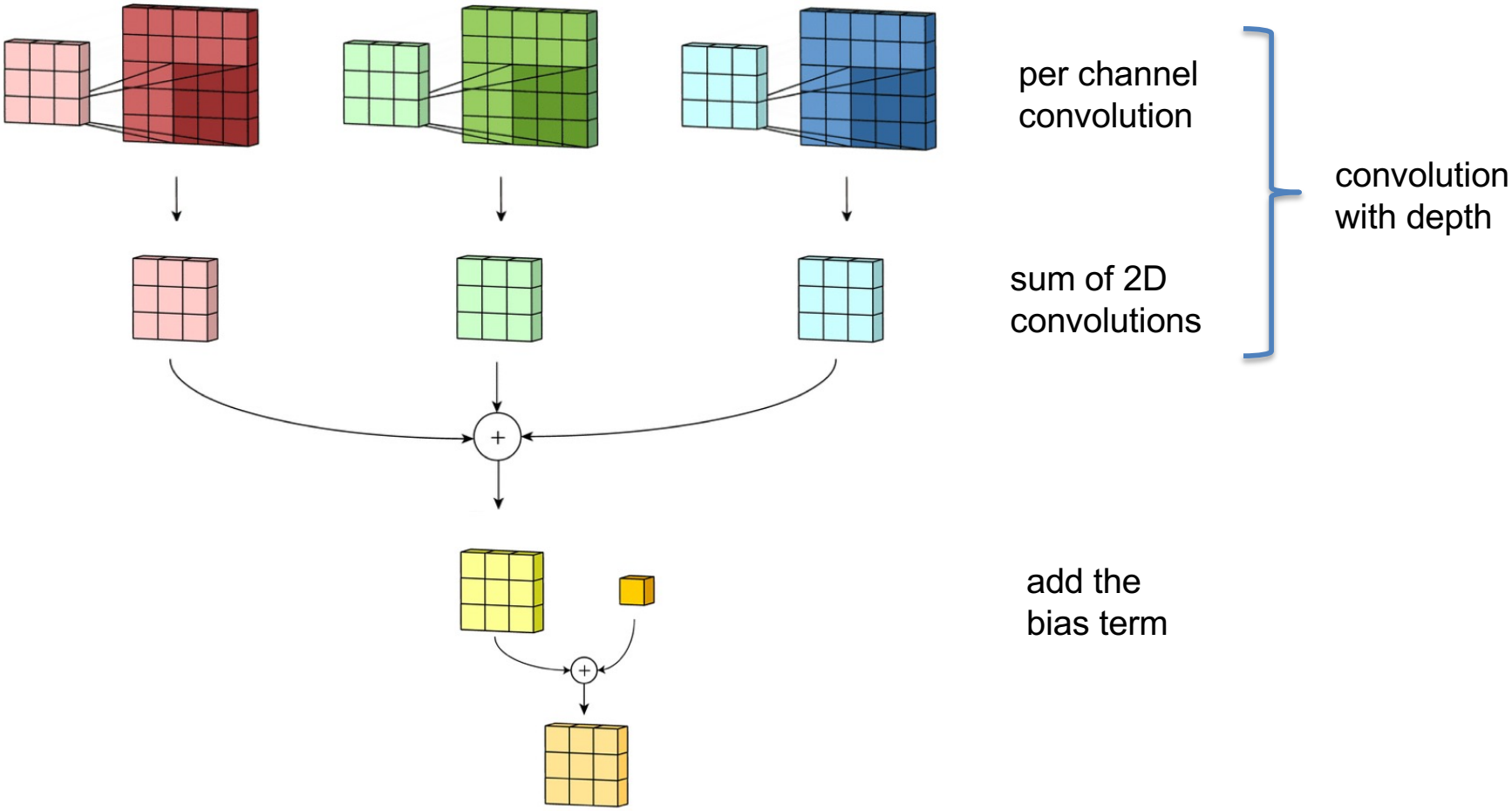
3_0	3_1	2_2	1	0
0_2	0_2	1_0	3	1
3_0	1_1	2_2	2	3
2	0	0	2	2
2	0	0	0	1

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

Source: https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

Convolution Layer

Channels of the input image (usually RGB) are represented as depth of a 3D block
→ Filters' outputs are also represented as channels in the depth dimension

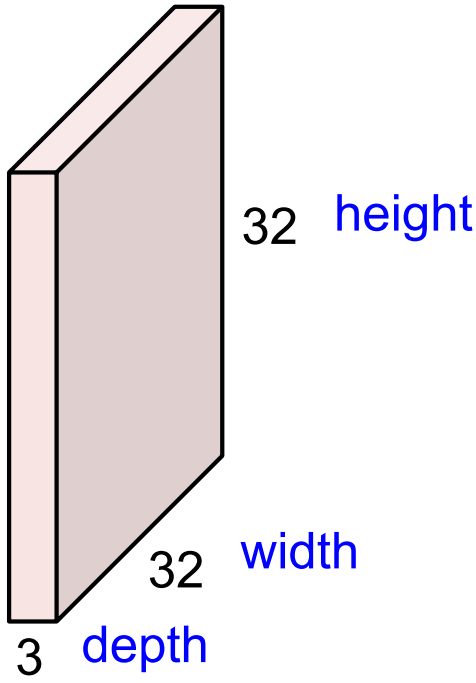


Source: https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

adapted from José Luis Alba

Convolution Layer

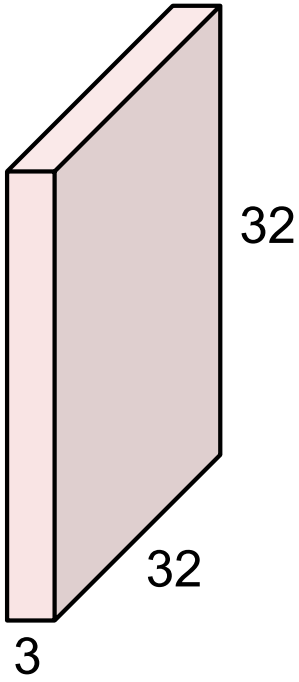
32x32x3 image



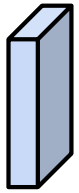
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image

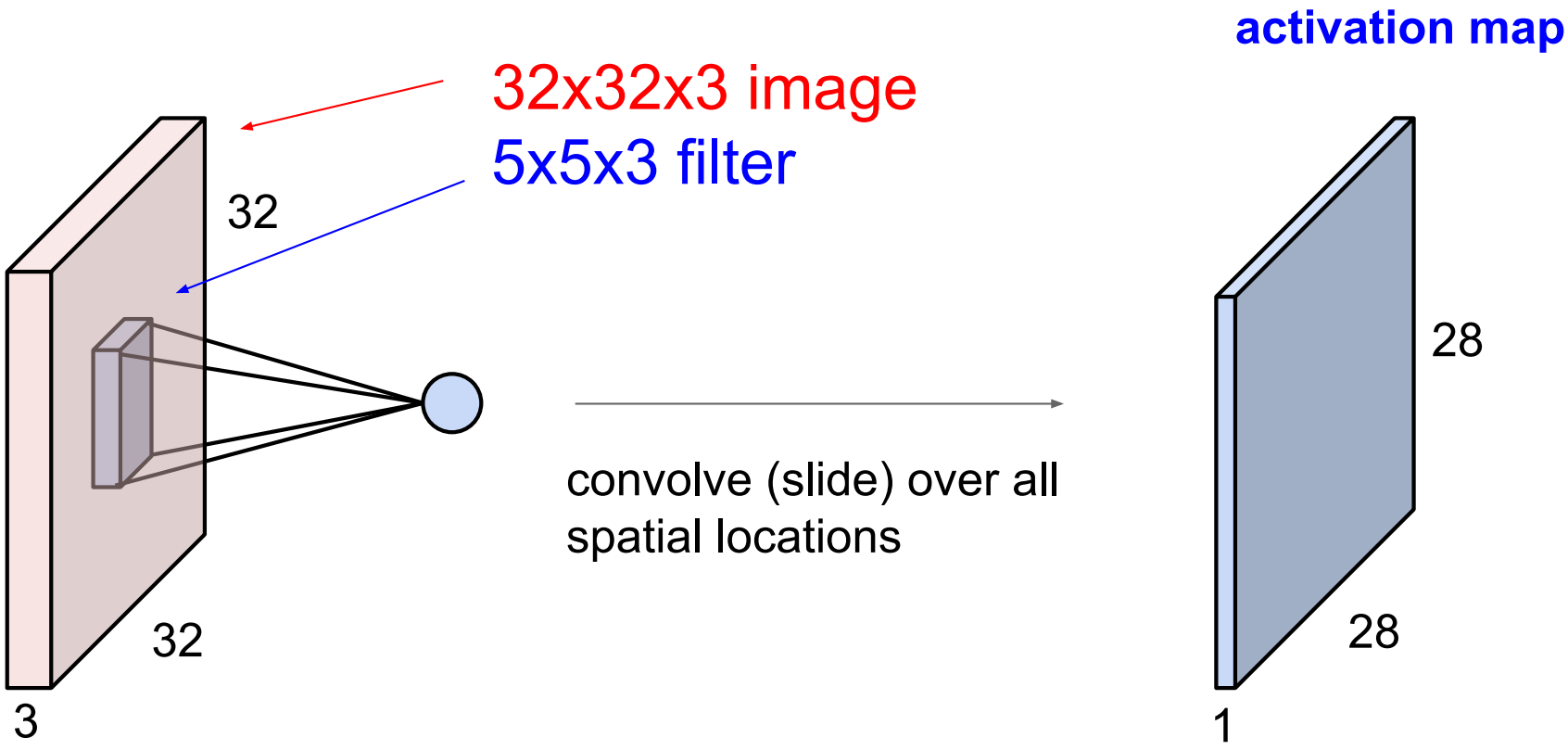


5x5x3 filter



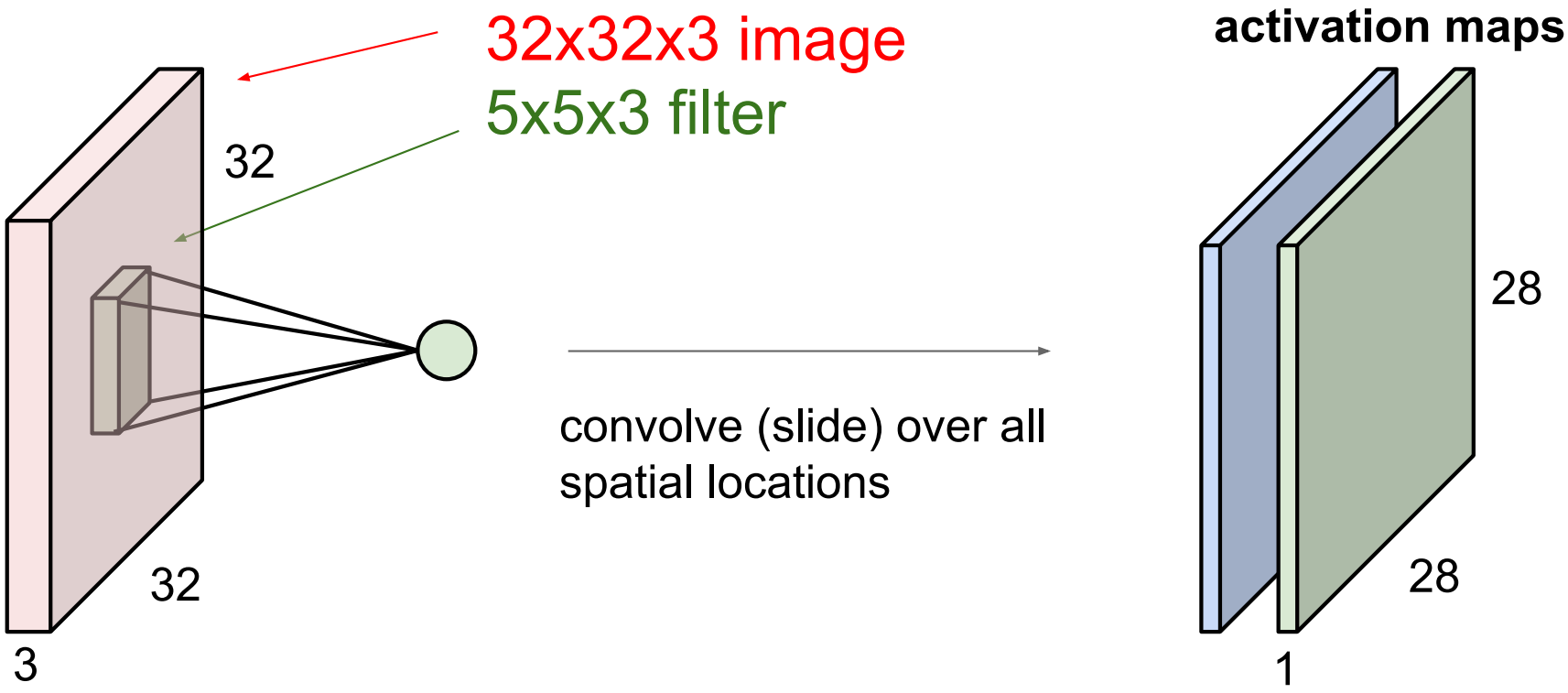
Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

Convolution Layer

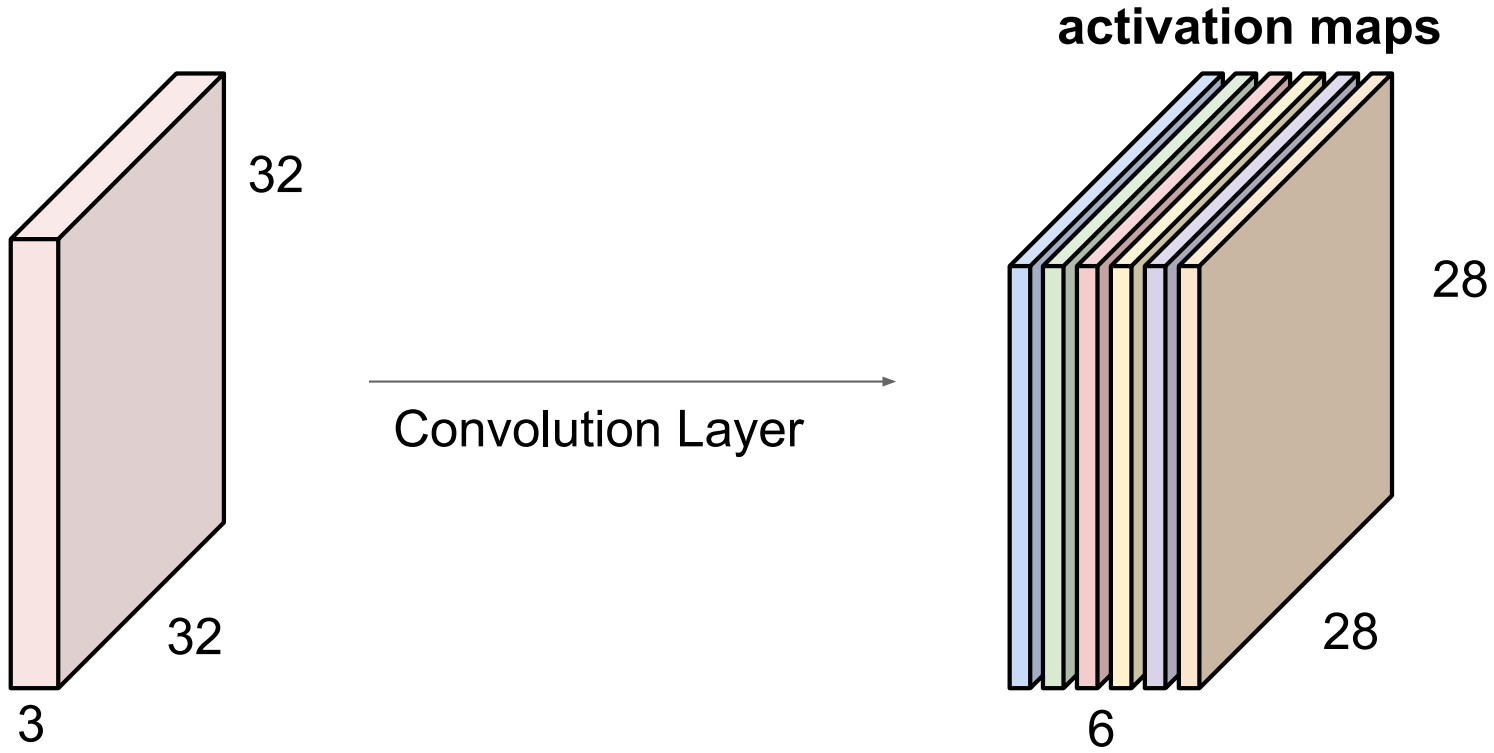


Convolution Layer

consider a second, green filter

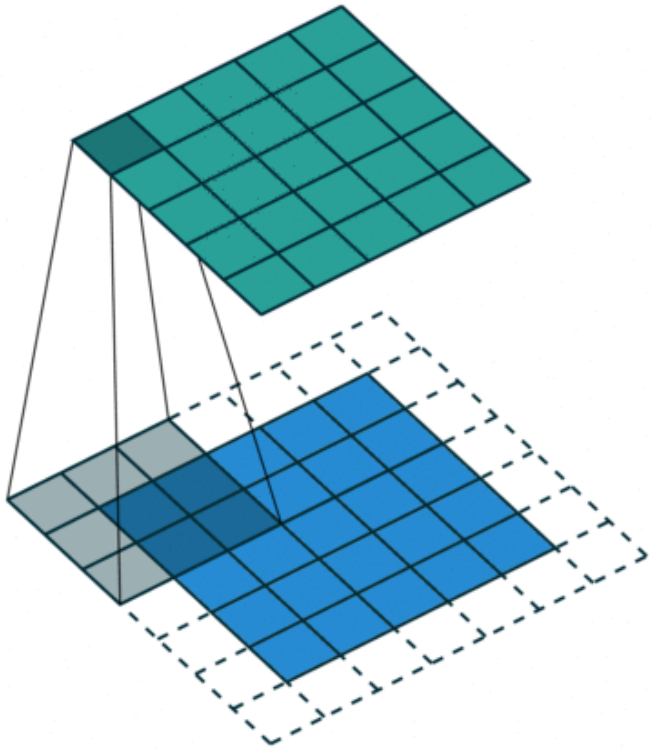


For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

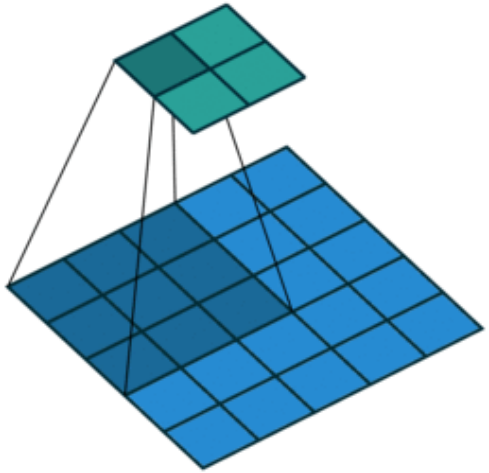


We stack these up to get a "new image" of size 28x28x6!

Padding and Stride



padding - avoids shrinking due to the border pixels



strided convolutions - use a step > 1 (usually the same for all dimensions, but mandatory)

Hyperparameters*

- **# of filters**: integer indicating the # of filters applied to each window
- **kernel size**: tuple (width, height) indicating the size of the window
- **stride**: tuple (horizontal, vertical) indicating the horizontal and vertical shift between each window.
- **padding**: boolean to indicate if the input is padded with a border of zeros to ensure that the output has the same size as the input.

*hyperparameter are used to control the learning process; parameters are learned

Exercise

- Calculate the **output volume size** {width, height, channels} of a convolutional layer composed of 10 filters of size 3x3 without padding and stride of 1, when applied to a color input image of size 6x6.
- Also, **how many parameters** (learnable) does this layer have?

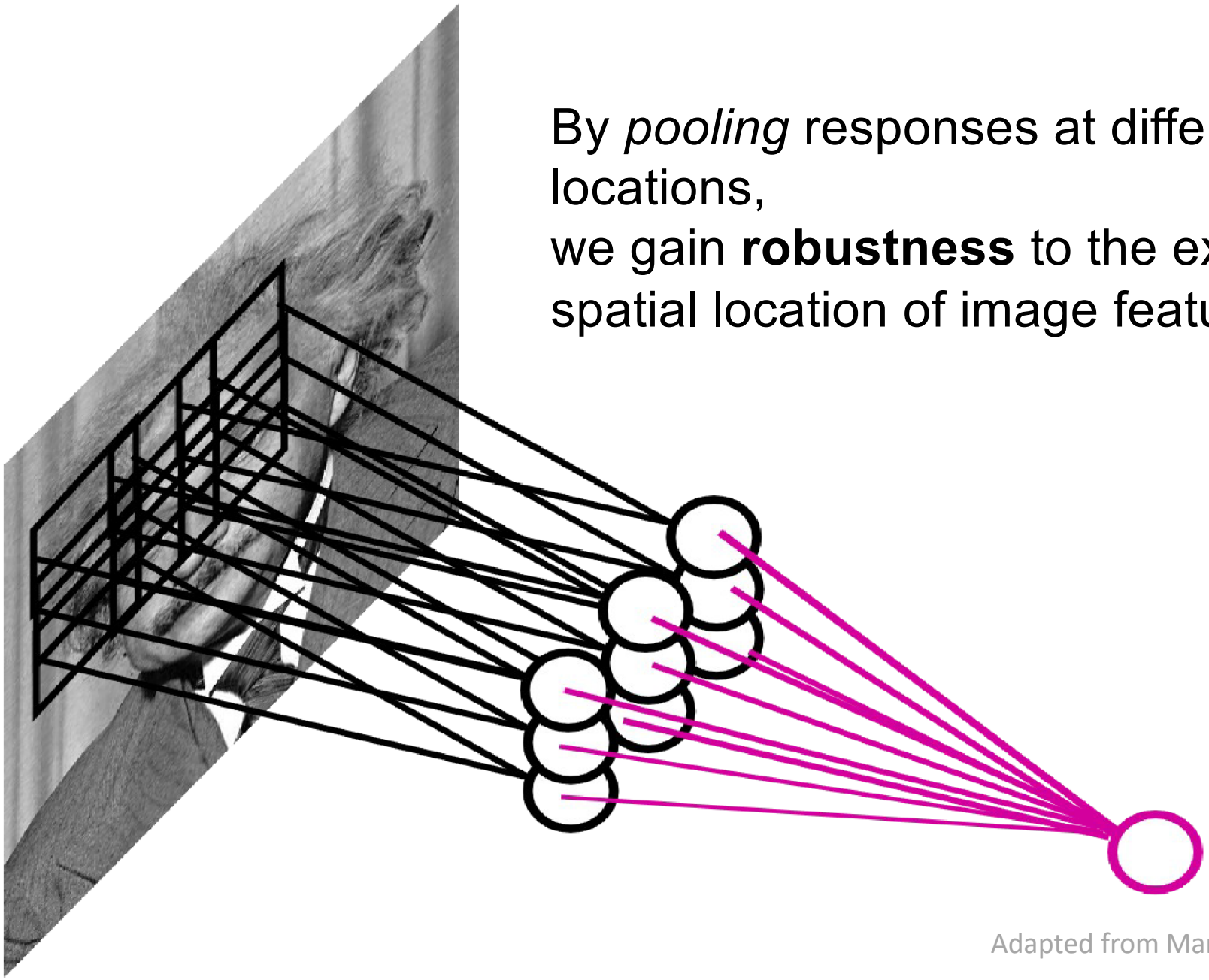
Pooling

- Pooling: **commutative** mathematical operation that combines several units
  - Examples: max, sum, product, average, Euclidean norm, etc.
  - Commutative property (order does not matter)

$$\max(a, b) = \max(b, a)$$

  - Makes the representations smaller and more manageable
  - More robust to location

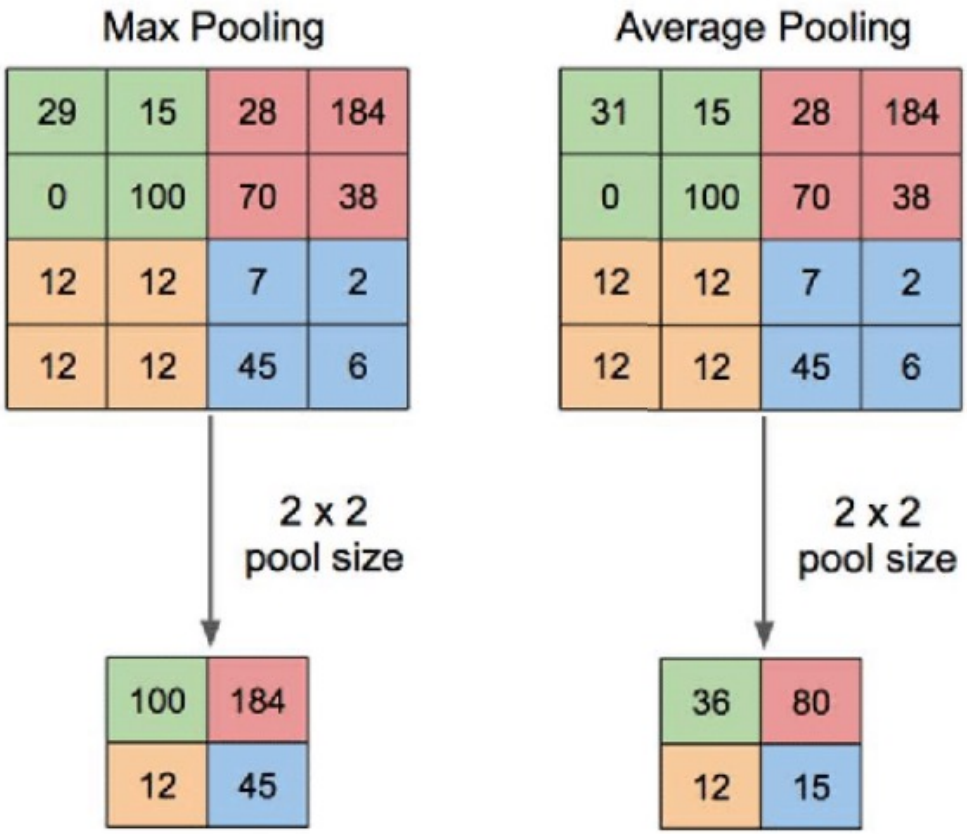
Pooling Layer



By *pooling* responses at different locations,
we gain **robustness** to the exact spatial location of image features.

Adapted from Marc'Aurelio Ranzato

Pooling Layer

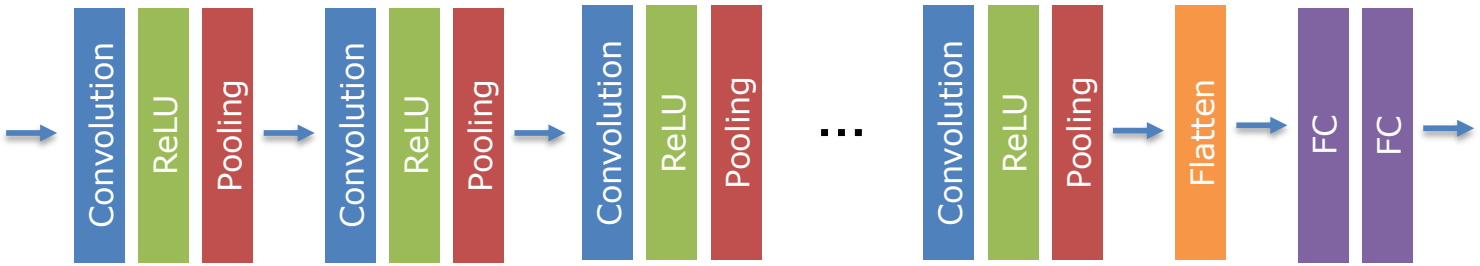


The operations are done per channel → same depth as input

How many parameters?
And how many hyperparameters?

Building blocks of a CNN

- Stacking Convolutional layers we can build a NN with several interesting properties.



- Inserting a non-linear activation function we achieve non-linear **complex transformations**.
- Inserting pooling layers we can down-sample the spatial information and **trade with more feature filters** learnt during training.
- Each new layer learns on top of previously filtered images with **less spatial information and more extracted features**.
- Inserting a final classifier (fully-connected MLP) over a "flattened" image allows to do classification or regression, not on the raw pixels but on **learnt features of the image**.

adapted from José Luis Alba

From Local to Global

- *The **receptive field** is defined as the region in the input space that a particular CNN feature is looking at (i.e. be affected by)*
- Increases as we go deeper in the network

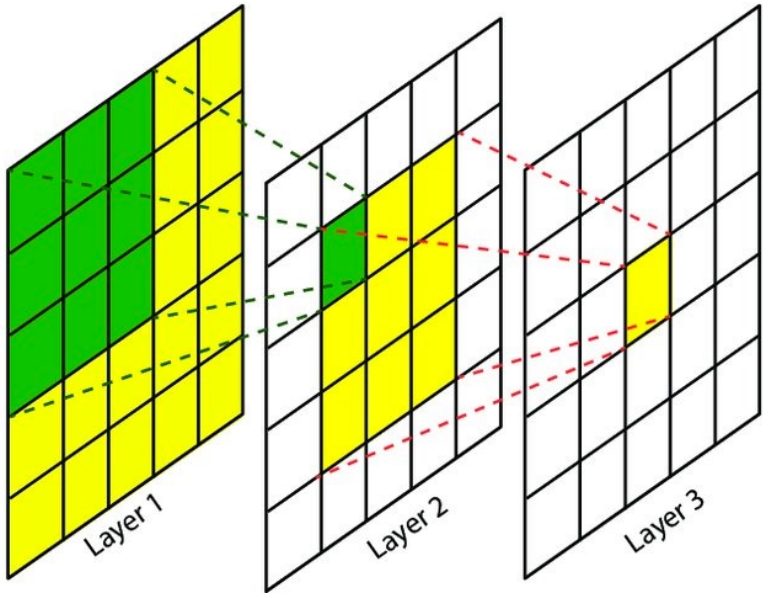


Image Source: 10.3390/rs9050480

CNN Summary

- High accuracy for image applications
- Special purpose net – for images or problems with strong local **spatial/temporal correlation**
- Lots of handcrafting and tuning to find the right recipe of receptive fields, layer interconnections, etc.
  - Lots more hyperparameters than standard nets, and even than other deep networks, since the **structures of CNNs** are so **handcrafted**
  - CNNs getting wider and deeper with speed-up techniques
- Training requires a **large amount of data**